What is claimed is:

1. An article comprising a machine readable medium storing instructions that, if executed by a machine, cause the machine to perform a plurality of operations comprising:

specifying a monitor address;

suspending a thread until a monitor break event occurs;

testing whether the monitor break event is a write to the monitor address;

if the monitor break event is not the write to the monitor address, then suspending the thread again.

2. The article of claim 1 wherein suspending the thread again comprises returning to specifying the monitor address.

3. The article of claim 2 wherein specifying the monitor address comprises executing a MONITOR instruction and wherein suspending the thread until the monitor break event occurs comprises executing an MWAIT instruction.

4. The article of claim 1 wherein said plurality of operations further comprise, after specifying the monitor address and before suspending the thread:

testing whether data at the monitor address has changed.

5. The article of claim 1 wherein specifying the monitor address comprises executing an

instruction with an operand chosen from a set consisting of a linear address, a virtual address, a physical address, and a relative address.

6. The article of claim 5 wherein the operand is one of a second set consisting of an explicit operand and an implicit operand.

7. The article of claim 1 wherein said monitor address specifies a cache line.

8. The article of claim 2 wherein said plurality of operations further comprise providing a second operand as a mask operand to control which events are monitor break events.

9. An article comprising a machine readable medium storing instructions that, if executed by a machine, cause the machine to perform operations comprising:

programming a monitor with a monitor address corresponding to a cache line of at least one work location;

suspending a thread until a monitor break event occurs;

testing whether the at least one work location indicates a first task is ready to execute;

testing whether the at least one work location indicates a second task is ready to execute;

if neither the first task nor the second task is ready to execute, then returning to suspending the thread.

10. The article of claim 9 wherein returning to suspending the thread until the monitor break event occurs further comprises re-programming the monitor with the monitor address prior to suspending the thread.

11. The article of claim 9 wherein returning to suspending the thread comprises returning to programming the monitor with the monitor address.

12. A method comprising:

specifying a monitor address;

suspending a thread until a monitor break event occurs;

testing whether the monitor break event is a write to the monitor address;

if the monitor break event is the write to the monitor address, then suspending the thread again.

13. The method of claim 12 wherein suspending the thread again comprises returning to specifying the monitor address.

14. The method of claim 13 wherein specifying the monitor address comprises executing a MONITOR instruction and wherein suspending the thread until the monitor break event occurs comprises executing an MWAIT instruction.

15. The method of claim 12 wherein said method further comprises, after specifying the monitor address and before suspending the thread:

testing whether data at the monitor address has changed

16. The method of claim 12 wherein specifying the monitor address comprises executing an instruction with an operand chosen from a set consisting of a linear address, a virtual address, a physical address, and a relative address.

17. The method of claim 16 wherein programming the operand is one of a second set consisting of an explicit operand and an implicit operand.

18. The method of claim 1 wherein said method further comprises enabling recognition of writes to the monitor address as monitor break events.

19. The method of claim 13 further comprising providing a second operand as a mask operand to control which events are monitor break events.

20. A system comprising:

a processor;

a monitor to generate a monitor break event in response to a memory access to a monitor address;

event detect logic to detect an of a plurality of monitor break events;

a memory to store a loop in a first thread executable by said processor to specify said monitor address and to repeatedly suspend said first thread after monitor break events until the memory access to the monitor address occurs.

21. The system of claim 20 wherein said loop comprises:

a first instruction to specify the monitor address;

a second instruction to suspend said first thread.

22. The system of claim 21 wherein said loop further comprises a test after said first instruction to determine whether data at the monitor address has changed after execution of the first instruction but before execution of the second instruction, wherein said loop exits without execution of the second instruction if data at the monitor address has changed.

23. The system of claim 21 wherein said loop further comprises a test after said first instruction to determine whether data at the monitor address has changed after execution of the second instruction wherein said loop performs another iteration if data at the monitor address has not changed.

24. The system of claim 20 wherein said loop comprises:

a test to determine whether a work location in a first cache line indicated by the monitor address contains a first value, wherein a first routine is executed if said work location contains the first value;

a second test to determine whether the work location in said first cache line contains a second value, wherein a second routine is executed if said work location contains the second value;

an instruction to suspend said first thread if said work location does not contain

said first value and said work location does not contain said second value.

25. A system comprising:

a processor;

a monitor;

a memory to store an idle loop in a first thread executable by said processor to perform operations comprising:

specifying a monitor address;

suspending said first thread until a monitor break event occurs;

testing whether the monitor break event is a write to the monitor address;

if the monitor break event is not the write to the monitor address, then returning to specifying the monitor address.

26. The system of claim 25 wherein specifying the monitor address comprises executing a first instruction and wherein suspending the thread until the monitor break event occurs comprises executing a second instruction.

27. The system of claim 25 wherein said operations further comprise, after specifying the monitor address and before suspending the thread:

testing whether data at the monitor address has changed.

28. A method comprising:

executing a first instruction in a first thread that specifies a monitor address;

executing a second instruction in said first thread to suspend said first thread until

a write access implicating said monitor address or an interrupt occurs;

executing a plurality of instructions in a second thread;

after the write access or the interrupt occurs, testing whether a data element associated with said monitor address has changed;

returning to executing the second instruction if the data element has not changed.

29. The method of claim 28 wherein returning to executing the second instruction comprises returning to executing the first instruction and continuing on to executing the second instruction.

30. The method of claim 28 further comprising testing whether the data element associated with said monitor address has changed prior to executing said second instruction.